

Neither a Borrower nor a Lender Be: The Dangers of Mobile Code

Barton P. Miller
bart@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
Madison, Wisconsin 53705
USA



© 2001 Barton P. Miller

January 21, 2001

Security Issues

Four Security Topics

1. Subverting running programs
It's easy; we have a nice toolkit.
2. Safety checking of binary programs
Given an interface spec and the machine code,
verify safety conditions.
3. Safe remote execution of my job
The Condor or Java applet scenario.
4. An infrastructure for safe mobile computing.
Make mobility easier, while allowing the sysadmin to
retain control.



© 2001 Barton P. Miller

- 2 -

Security Issues

Dynamic Instrumentation

- ❑ Does not require recompiling or relinking
 - Saves time: compile and link times are significant in real systems.
 - Can instrument without the source code (e.g., proprietary libraries).
 - Can instrument without linking (relinking is not always possible).
- ❑ Instrument optimized code.



© 2001 Barton P. Miller

- 3 -

Security Issues

Dynamic Instrumentation (con'd)

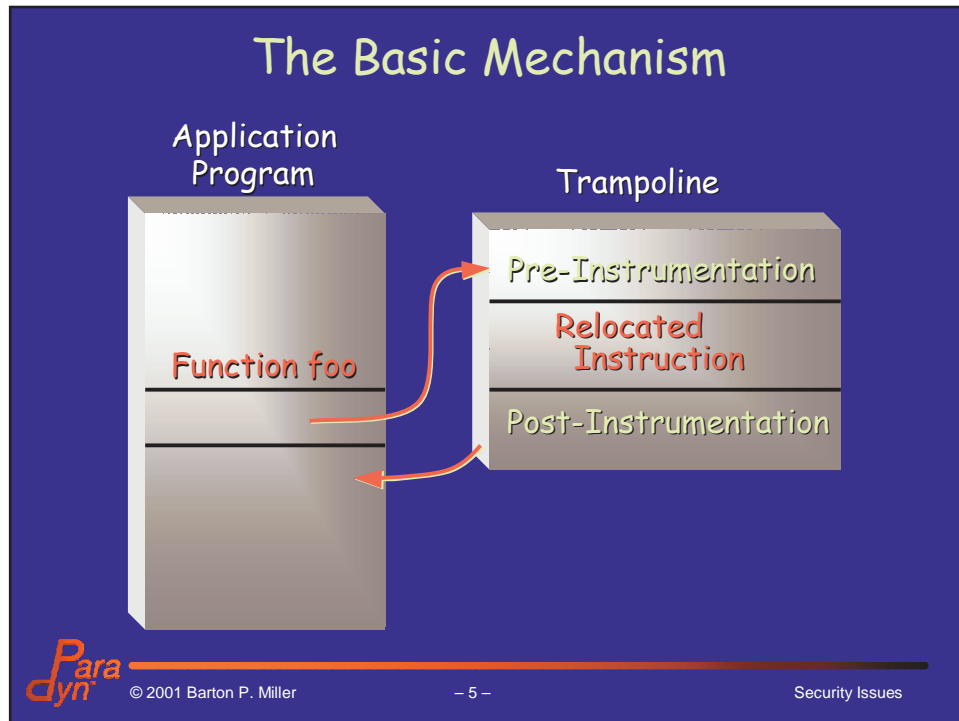
- ❑ Only instrument what you need, when you need
 - No hidden cost of latent instrumentation.
 - Enables "one pass" tools.
- ❑ Can instrument running programs (such as Web or database servers)
 - Production systems.
 - Embedded systems.
 - Systems with complex start-up procedures.



© 2001 Barton P. Miller

- 4 -

Security Issues



The DynInst Interface

- ❑ Machine independent representation
- ❑ Object-based interface to build Abstract Syntax Trees (AST's)
- ❑ Write-once, instrument-many (portable)
- ❑ Hides most of the complexity in the API
 - Process Hijacker: only 700 lines of user code!
 - MPI tracer: 250 lines

Para dyn

© 2001 Barton P. Miller

- 6 -

Security Issues

Basic DynInst Operations

- ❑ Process control:
 - Attach/create process
 - Monitor process status changes
 - Callbacks for fork/exec/exit
- ❑ Image (executable program) routines:
 - Find procedures/modules/variables
 - Call graph (parent/child) queries



© 2001 Barton P. Miller

- 7 -

Security Issues

Basic DynInst Operations

- ❑ Inferior (application processor) operations:
 - Malloc/free
 - Allocate heap space in application process
 - Inferior RPC
 - Asynchronously execute a function in the application.
 - Load module
 - Cause a new .so/.dll to be loaded into the application.



© 2001 Barton P. Miller

- 8 -

Security Issues

Basic DynInst Operations

□ Inferior operations (continued):

- Remove Function Call
 - Disable an existing function call in the application
- Replace Function Call
 - Redirect a function call to a new function
- Replace Function
 - Redirect all calls (current and future) to a function to a new function.



© 2001 Barton P. Miller

- 9 -

Security Issues

Basic DynInst Operations

□ Building AST code sequences:

- Control structures: if and goto
- Arithmetic and Boolean expressions
- Get PID/TID operations
- Read/write registers and global variables
- Read/write parameters and return value
- Function call



© 2001 Barton P. Miller

- 10 -

Security Issues

Applications of DynInst

- ❑ Process Hijacking (Vic Zandy)
 - Submitting already-running jobs to Condor
- ❑ MPI Tracer (Chris Chambreau)
 - Insert Vampir or Pablo trace calls on the fly.
- ❑ Function Call Tracer (Roland Wismüller)
 - Generate dynamic call graph
- ❑ Image Mentor (Brian Wylie)
 - Query module/function/memory structure
- ❑ Re-Tee (Jeff Hollingsworth)
 - Redirect program output on-the-fly
- ❑ License server bypassing
- ❑ Condor security attacks



© 2001 Barton P. Miller

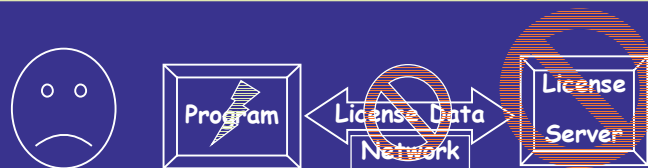
- 11 -

Security Issues

License Server Attack: The Bypass



Normal: licensed program runs after communicates with license server.



Undesired: licensed program refuses to run if license server does not respond.



© 2001 Barton P. Miller

- 12 -

Security Issues

Example: Adobe FrameMaker

Two-step license verification:

- retrieve license data from server [once]
- check license data for correctness [often]

In practice:

- allow FM to time-out waiting for server
- allow FM to attempt to go into "demo" mode
- switch FM back to normal mode
- insure that future license checks always succeed



© 2001 Barton P. Miller

- 13 -

Security Issues

Strategies

☐ Complete reverse engineering:

- not an option
 - legal problems
 - complexity (FrameMaker is a 7 MB binary!)

☐ Focus on certain characteristics:

- I/O (network sockets) traffic
- execution trace



© 2001 Barton P. Miller

- 14 -

Security Issues

Tools

- ❑ High-level language translators:
 - Dyners: interactive, interpreted C subset
 - Jdyninst: Java to DynInst compiler
- ❑ Bypasser: an interactive call graph browser
 - Search and walk application call graph
 - Resolves function pointers at runtime
 - Call follow caller or callee paths
 - Can generate call trace



© 2001 Barton P. Miller

– 15 –

Security Issues

Use

- ❑ Determining where to apply changes:
 - get trace for a successful run
 - get trace for a (forced-)failure run
 - compare to find differences
 - repeat as needed



© 2001 Barton P. Miller

– 16 –

Security Issues

Details

- ❑ FM calls `NlOpenlicenses` on start up
 - Contacts license server and caches credential if successful
- ❑ At end of main, and calls `NluiCheckLicense`
 - If credential is not present, call `ChangeProductToDemo` (cannot save files)
- ❑ Frequently, during operation, FM will check for cached credentials.



© 2001 Barton P. Miller

- 17 -

Security Issues

Details

- ❑ FM calls `NlOpenlicenses` on start up
 - Contacts license server and caches credential if successful
 - Allow this to fail.
- ❑ At end of main, and calls `NluiCheckLicense`
 - If credential is not present, call `ChangeProductToDemo` (cannot save files)
- ❑ Frequently, during operation, FM will check for cached credentials.



© 2001 Barton P. Miller

- 18 -

Security Issues

Details

- ❑ FM calls `NlOpenlicenses` on start up
 - Contacts license server and caches credential if successful
 - Allow this to fail.
- ❑ At end of main, and calls `NluiCheckLicense`
 - If credential is not present, call `ChangeProductToDemo` (cannot save files)
 - Delete the call to `ChangeProductToDemo`.
- ❑ Frequently, during operation, FM will check for cached credentials.



© 2001 Barton P. Miller

- 19 -

Security Issues

Details

- ❑ FM calls `NlOpenlicenses` on start up
 - Contacts license server and caches credential if successful
 - Allow this to fail.
- ❑ At end of main, and calls `NluiCheckLicense`
 - If credential is not present, call `ChangeProductToDemo` (cannot save files)
 - Delete the call to `ChangeProductToDemo`.
- ❑ Frequently, during operation, FM will check for cached credentials.
 - Change this call to always return "true".



© 2001 Barton P. Miller

- 20 -

Security Issues

Condor Attack: Lurking Jobs

- ❑ Condor schedules jobs on idle workstations
- ❑ In a normal mode, jobs run as a common, low-privilege user ID: "nobody".
- ❑ This common user ID provides an opportunity for an evil lurking process to ambush subsequent jobs (from other users):

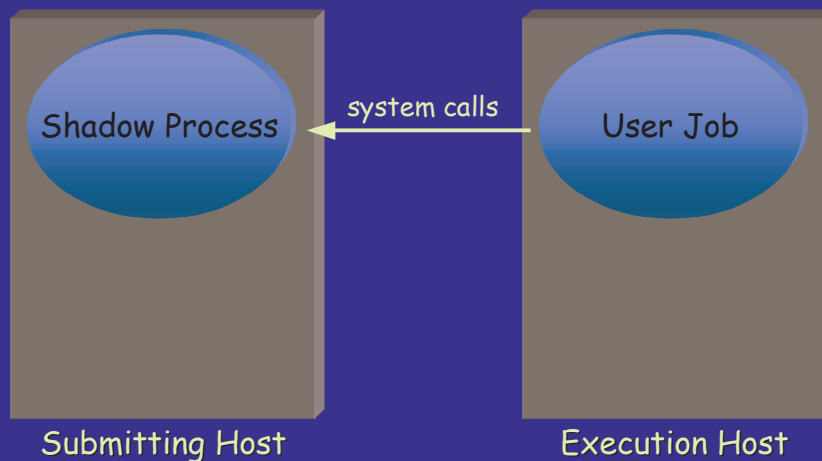


© 2001 Barton P. Miller

- 21 -

Security Issues

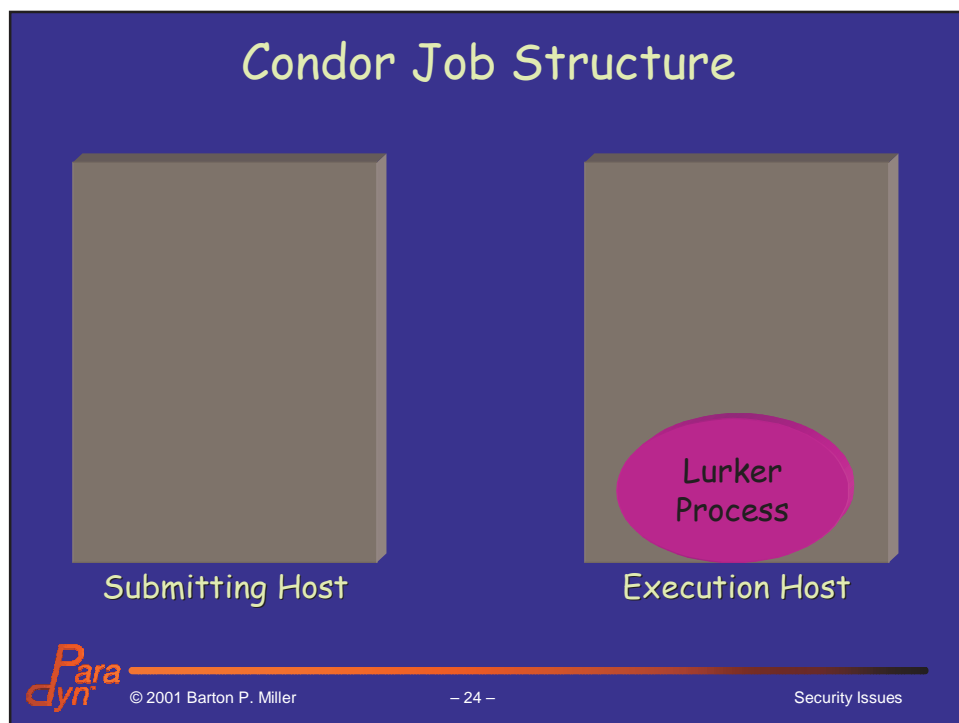
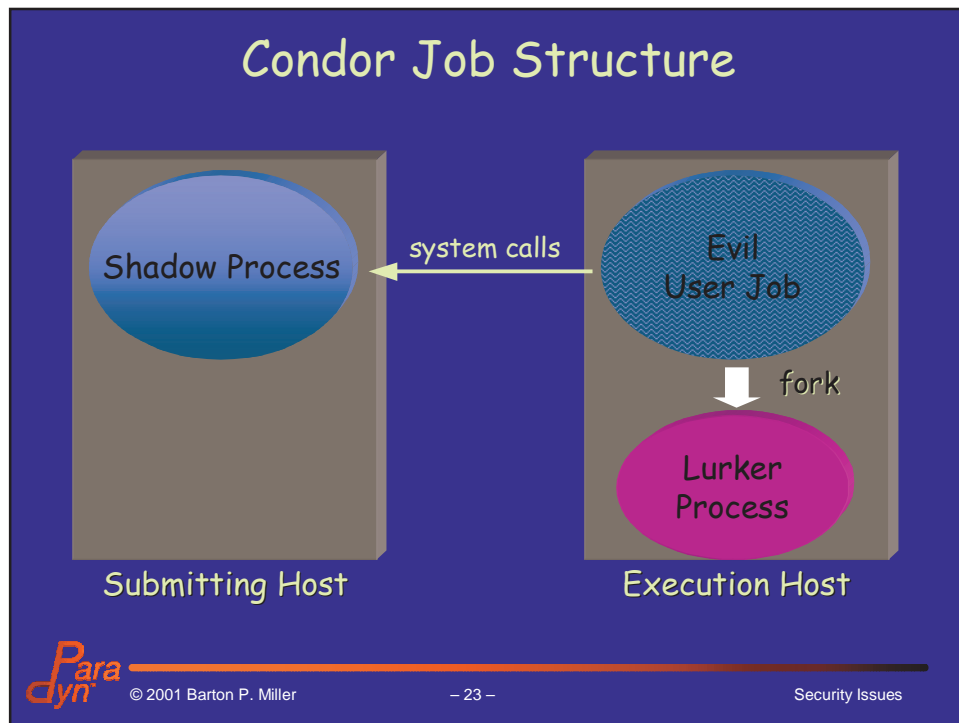
Condor Job Structure

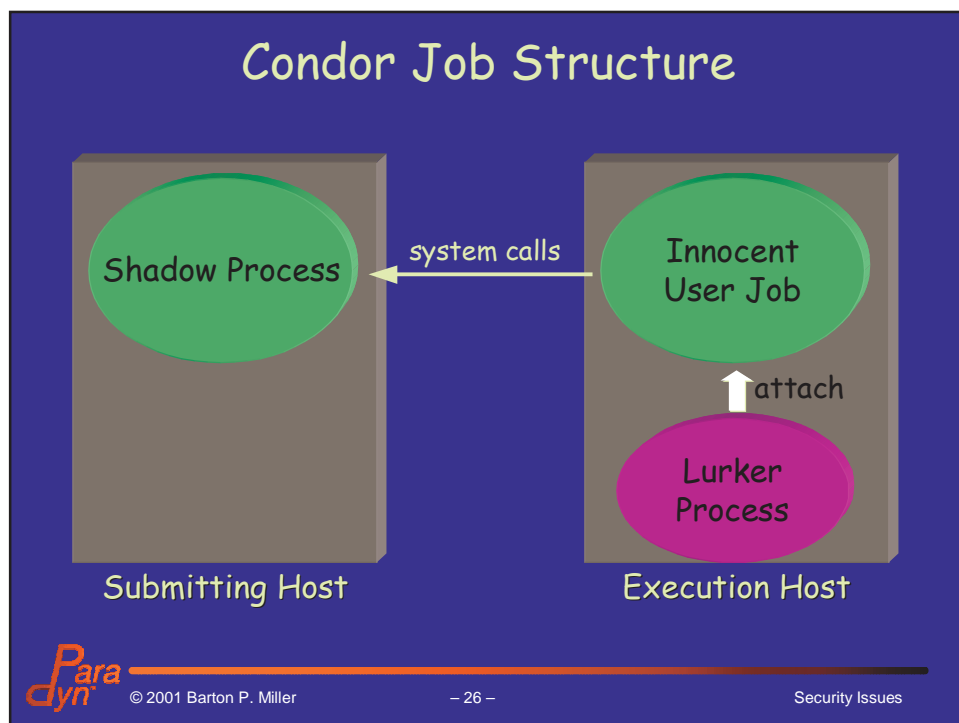
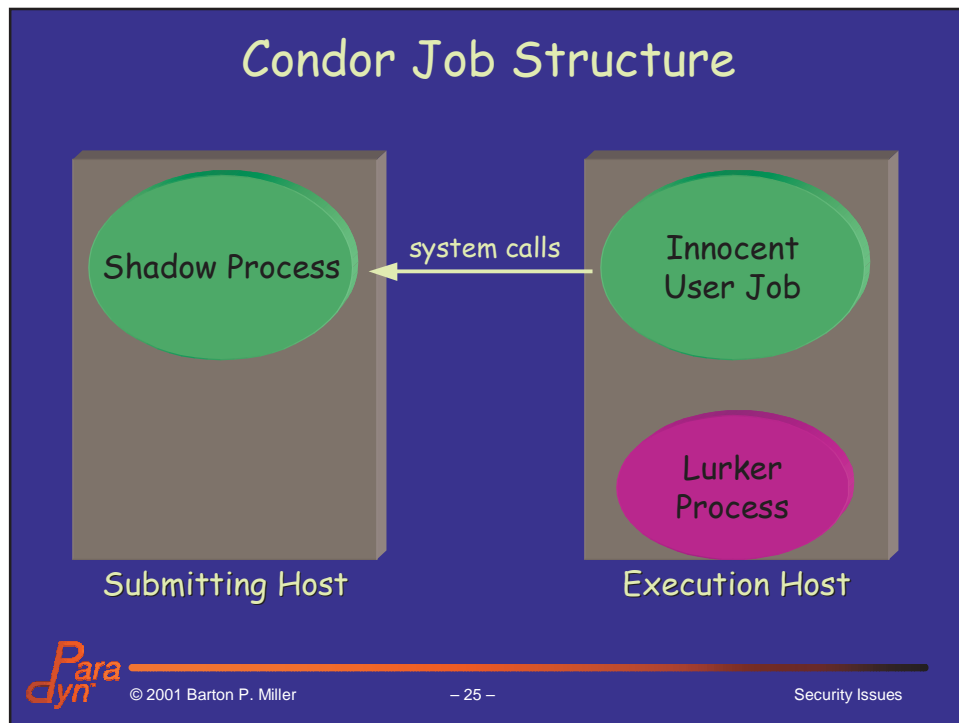


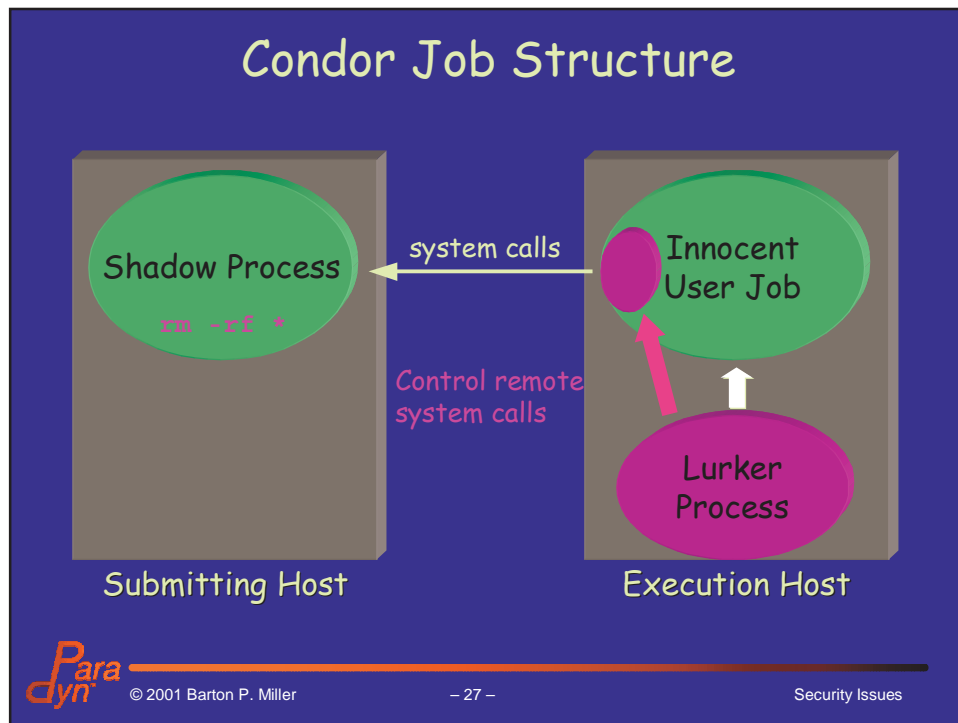
© 2001 Barton P. Miller

- 22 -

Security Issues



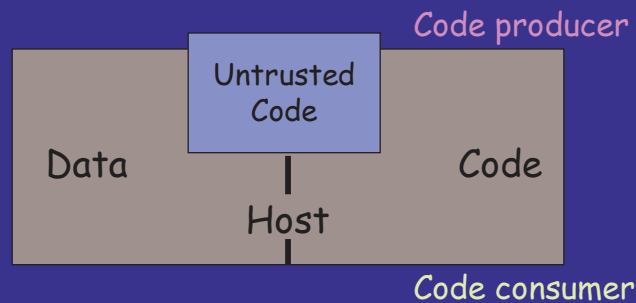




Copyright © 1999 United Feature Syndicate, Inc.
Redistribution in whole or in part prohibited

2. Safety Checking of Binary Code

- Is it safe for **untrusted foreign code** to be loaded into a trusted **host system**?



Para
dyn

© 2001 Barton P. Miller

- 29 -

Security Issues

Safety Properties We Enforce

- Default collection of safety conditions
 - No type violations
 - No out-of-bounds array accesses,
 - No misaligned loads/stores,
 - No uses of uninitialized variables,
 - No invalid pointer dereferences,
 - No unsafe interaction with the host

Type safety
- Precise and flexible host access policy
 - Customizable

Para
dyn

© 2001 Barton P. Miller

- 30 -

Security Issues

Motivation

□ Dynamic extensibility

- Operating systems: custom policies, general functionality, performance
 - Extensible OS: exokernel, VINO, SPIN, synthetix...
 - Commodity OS: SLIC, kerninst, ...
- Databases: type-based extensions
 - Illustra, informix, paradise, ...
- Web browsers: plug-ins
- Performance tools: measurement code
 - Kerninst, paradyn, ...
- Active network components



© 2001 Barton P. Miller

- 31 -

Security Issues

Motivation

□ Component-based software (Java, COM)

- Software components from different vendors are combined to construct a complete application
- Code from several sources with no mutual trust



© 2001 Barton P. Miller

- 32 -

Security Issues

High-Level Characteristics

- ❑ Perform safety checking on ordinary binary, mechanically synthesize (and verify) a safety proof
- ❑ Extend the host at a very fine-grained level (allow the untrusted code to manipulate the internal data structures of the host directly)
- ❑ Enforce host-specified access policy + type safety



© 2001 Barton P. Miller

- 33 -

Security Issues

Host-Specified Access Policy

- ❑ Classify locations into regions
As big as the entire address space, as small as a variable
 - ❑ [Region : Category : Access]
 - Category: Types, fields
 - Access:
 - readable (r),
 - writable (w),
 - followable (f),
 - executable (e),
 - operable (o)
- Locations
- Values
- (e.g., to "copy", to "examine")



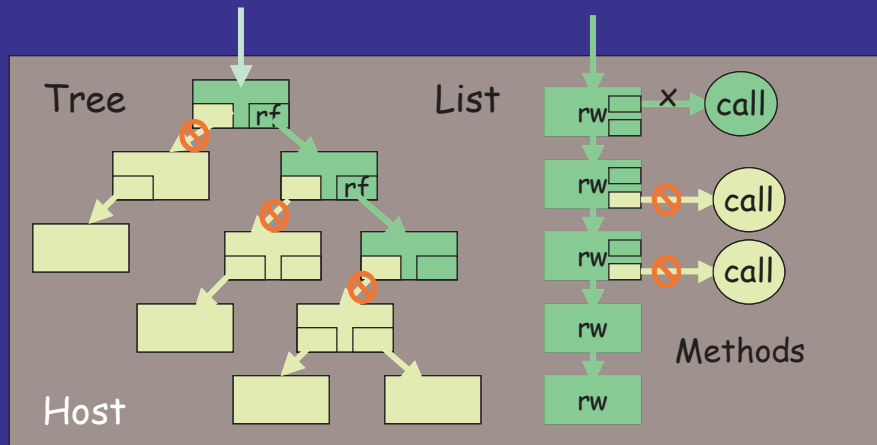
© 2001 Barton P. Miller

- 34 -

Security Issues

Protections Provided by Access Policy

Initial inputs to the untrusted code



Para
dyn

© 2001 Barton P. Miller

- 35 -

Security Issues

Principle of "Least Privilege"

□ Kernel page-replacement extension

- Pick a cold page from global LRU list.

```
typedef struct _page_list {  
    int page;                // read access  
    struct _page_list * next; // follow access  
} page_list;
```

[Host : page_list.page : ro]

[Host : page_list.next, page_list ptr : rfo]

Para
dyn

© 2001 Barton P. Miller

- 36 -

Security Issues

3. Safe Remote Execution of My Job

My job is executing on a remote host of unknown pedigree.

Threats:

- Can I trust the requests that are being made from the remote job to my home host?
- Can I trust the results that are being calculated by the remote job?

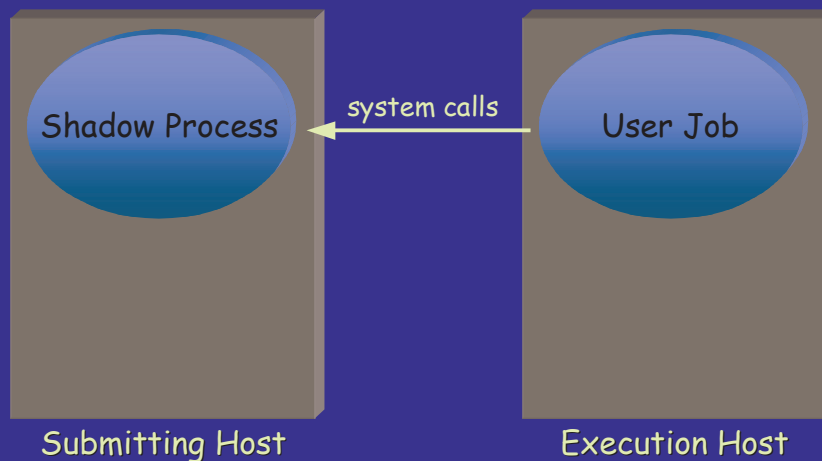


© 2001 Barton P. Miller

- 37 -

Security Issues

Condor Job Structure



© 2001 Barton P. Miller

- 38 -

Security Issues

Three Approaches

1. Filtering: screen out dangerous requests

- Sandboxing: restrict particular syscalls, do a chroot, restrict host access.
- Behavioral profiling, ala intrusion detection: use past behavior to screen future requests.

This technique addresses the threat to the home host, but not the data integrity problem.



© 2001 Barton P. Miller

– 39 –

Security Issues

Three Approaches

2. Replication

- Byzantine-like replication to detect and tolerate malicious modifications.
- Similar techniques to detect and tolerate malicious remote requests.

Addresses both threats, but at a high cost.



© 2001 Barton P. Miller

– 40 –

Security Issues

Three Approaches

3. "Slippery" jobs and "Crystal" jobs

- Design the program/process so that it is hard to get a handle:
 - System defensive techniques from worm technology.
 - Code transformations to keep the code unrecognizable.
- Slippery: cannot get a meaningful hold on the job.
- Crystal: is doesn't bend, but it shatters
 - Any modification is likely to destroy, rather than subvert the job.

This area is in the crazy-idea stage. Stay tuned!



© 2001 Barton P. Miller

- 41 -

Security Issues

4. Ubiquitous Mobility

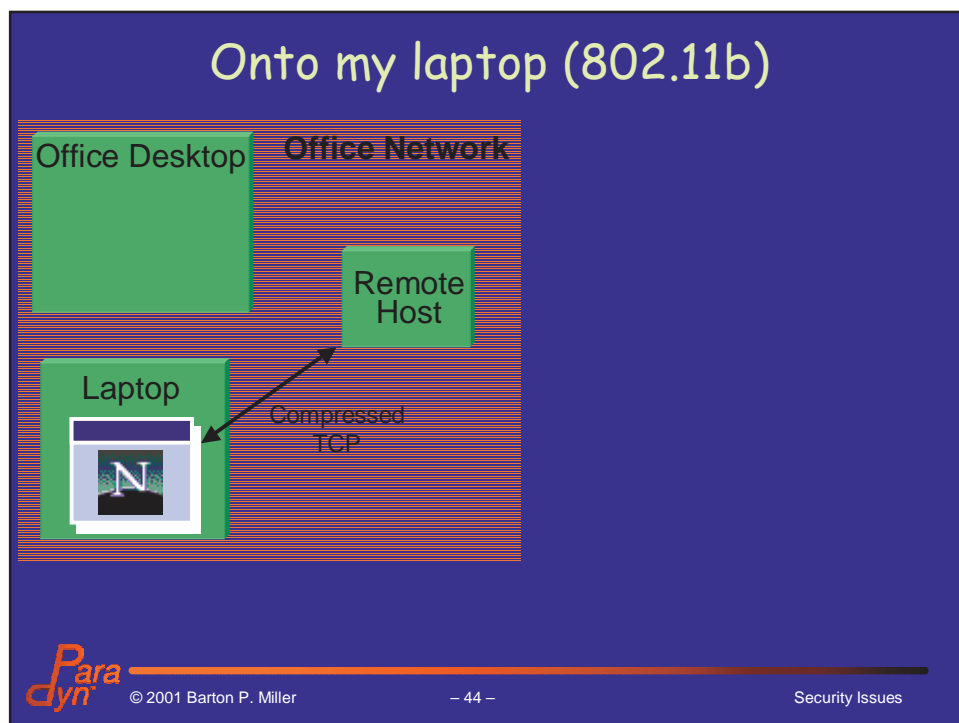
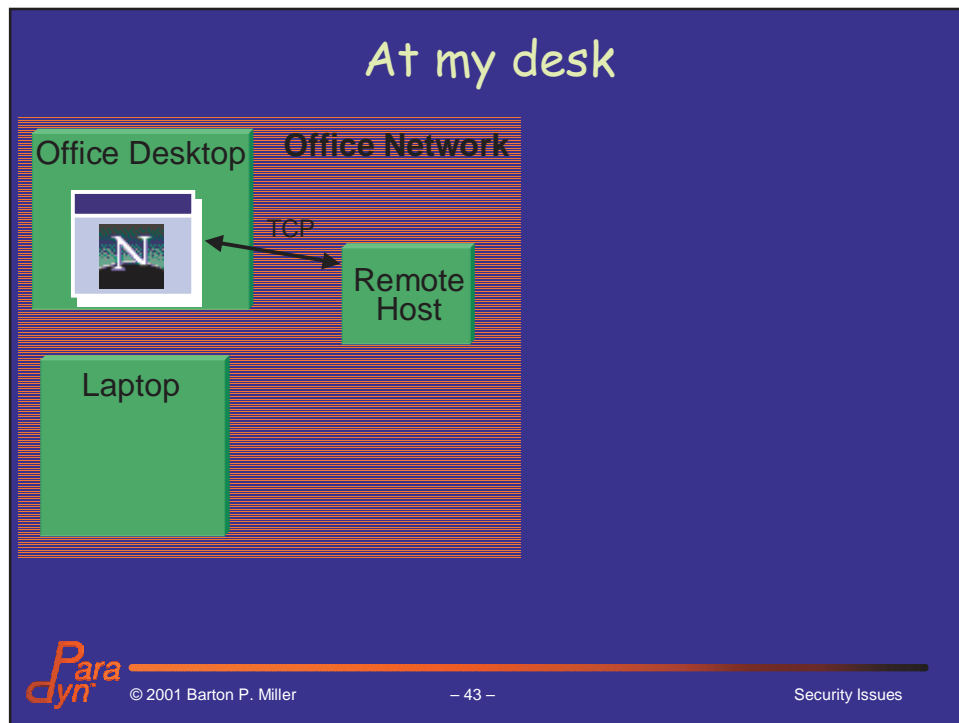
- ☐ Ordinary applications, in execution, that move as:
 - User moves to a new computer
 - Computer moves to a new location
- ☐ No modifications to apps, OS, or network
 - Built on common existing infrastructure
- ☐ Security policy set by administrator, not user
- ☐ Everything is mobile
 - Network connections, GUIs, I/O

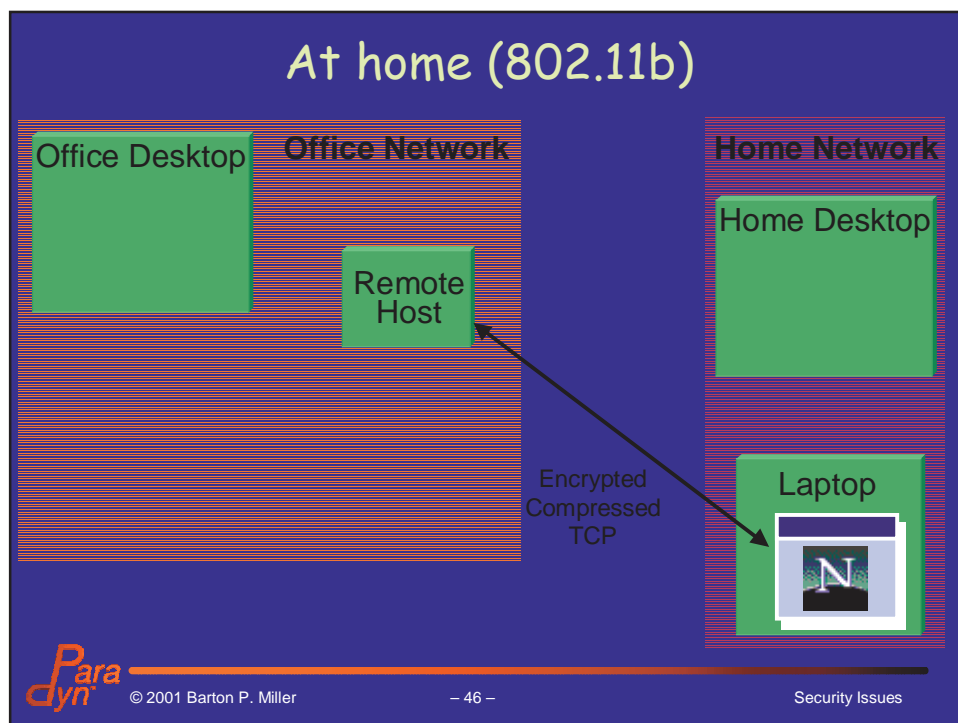
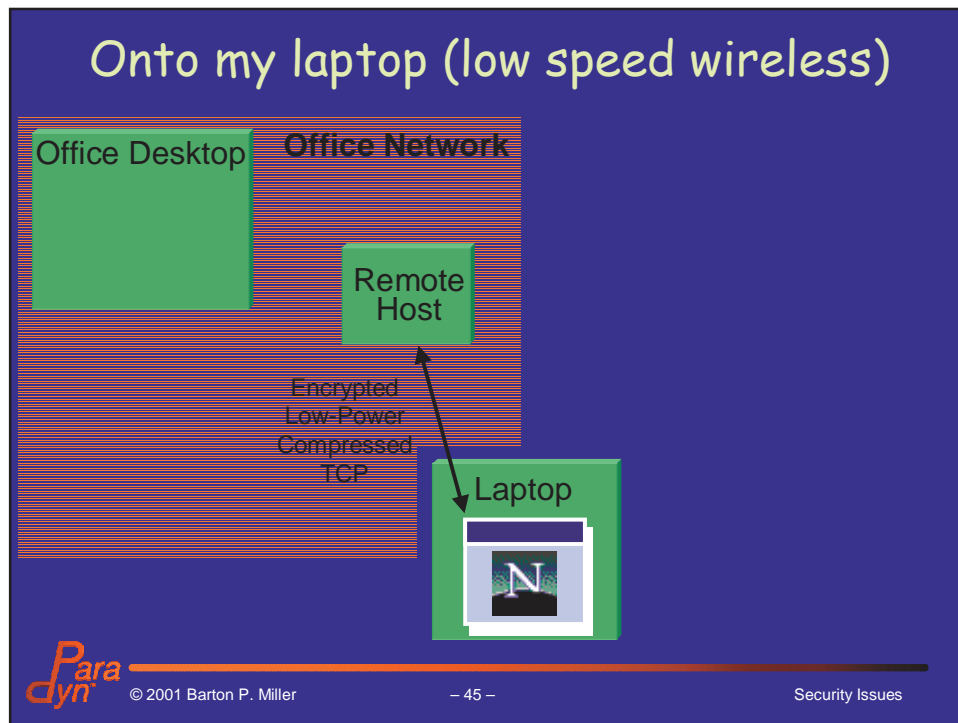


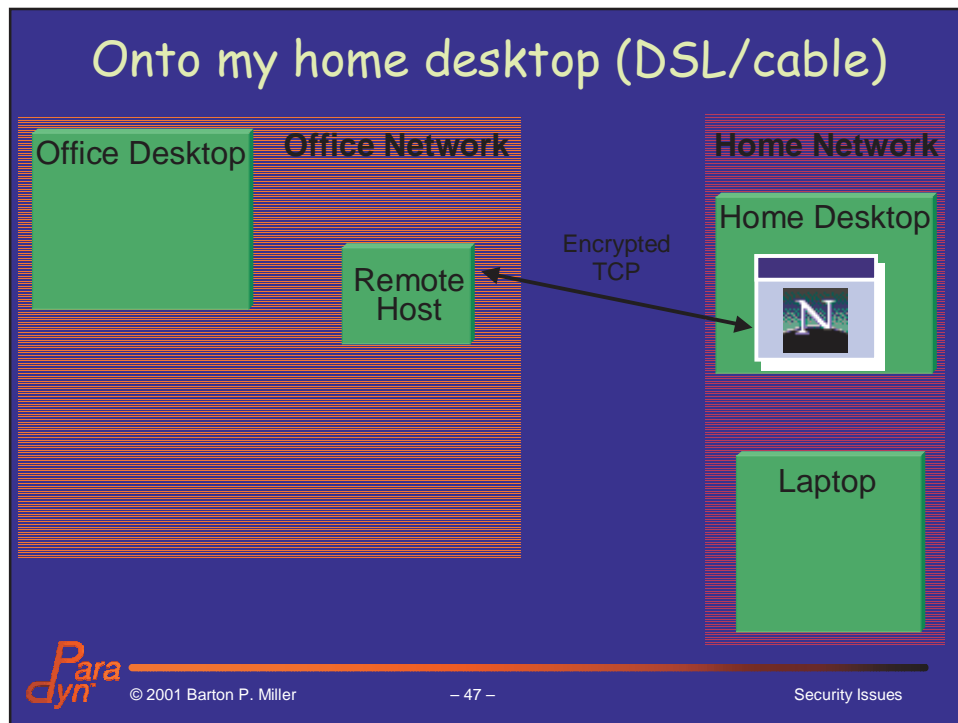
© 2001 Barton P. Miller

- 42 -

Security Issues







Components for Ubiquitous Mobility

- ☐ **Reliable Sockets**
 - Mobile, fault-tolerant network connections
- ☐ **Network Proxy**
 - Secure and mobile network connections with unmodified correspondents
- ☐ **GUI Proxy**
 - Mobile user interfaces
- ☐ **I/O Proxy**
 - Remote file access, based on Condor shadow

Para dyn © 2001 Barton P. Miller – 48 – Security Issues

Follow-Ups:

Dyninst Release 1.3:

- Runs on Solaris (SPARC & x86), Windows NT, AIX/SP2, Linux (x86), Irix (MIPS), Tru64 Unix.

<http://www.cs.wisc.edu/paradyn>

- Papers on dyninst, process hijacking, static safety checking:

<http://www.cs.wisc.edu/paradyn/papers>



© 2001 Barton P. Miller

– 49 –

Security Issues

Back-Up Slides

Static safety checking of binary code



© 2001 Barton P. Miller

– 50 –

Security Issues

Experimental Evaluation

□ Test cases

Array sum, start/stop timer, b-tree, kernel paging policy, hash, bubble sort, heap sort, stack-smashing, MD5, jPVM, /dev/kerninst (symbol, loggedWrites)

□ Summary of Results

- Found safety violations in kernel policy, stack-smashing, /dev/kerninst
- Verified all conditions, except for some calls in MD5, jPVM (precision lost due to inability to detect that a loop 'kills' all elements of an array)
- Checking times vary from 0.1 to 30 seconds



© 2001 Barton P. Miller

- 51 -

Security Issues

Characteristics of Test Cases

	Sum	Paging Policy	Start Timer	Hash	Bubble Sort	Stop Timer	Btree	Btree2	Heap Sort 2	Heap Sort	Stack-smashing	jPVM	/dev/kerninst /symbol	/dev/kerninst /loggedWrites	MD5
Instructions	13	20	22	25	25	36	41	51	71	95	309	315	339	358	883
Branches	2	5	1	4	5	3	11	11	9	16	89	16	45	36	11
Loops (Inner)	1	2(1)	0	1	2(1)	0	2(1)	2(1)	4(2)	4(2)	7(1)	3	6(4)	6	5(2)
Procedure Calls (Trusted)	0	0	1(1)	1	0	2(2)	0	4(4)	3	0	2	40(40)	36(25)	48(12)	6
Global Safety Conditions (Bounds Checks)	4(2)	9	13	15(2)	16(8)	17	35(14)	39(14)	56(26)	84(42)	100(74)	99(18)	116(42)	192(40)	121(30)
Source Language	C	C	C	C	C	C	C	C	C	C	C	C in C++ style	C++	C++	C




© 2001 Barton P. Miller

- 52 -

Security Issues

Timing (Seconds)

	Sum	Paging Policy	Start Timer	Hash	Bubble Sort	Stop Timer	Btree	Btree2	Heap Sort 2	Heap Sort	Stack-smashing	jPVM	/dev/kerninst /symbol	/dev/kerninst /loggedWrites	Mod5
Typestate Propagation	0.02	0.05	0.02	0.04	0.04	0.03	0.09	0.11	0.17	0.15	0.69	3.05	4.88	15.4	5.92
Annotation	0.003	0.005	0.005	0.006	0.005	0.007	0.008	0.01	0.015	0.015	0.03	0.069	0.068	0.26	0.082
Range Analysis	0.01	0	0	0.01	0.03	0	0.03	0.04	0.08	0.12	0.54	0.24	0.68	0.95	1.24
Induction-Iteration	0.08	0.18	0.13	0.40	0.18	0.14	0.40	0.035	1.15	2.46	12.74	1.55	8.60	12.33	3.41
TOTAL	0.1	0.23	0.16	0.46	0.26	0.18	0.53	0.51	1.42	2.75	14.0	4.91	14.2	28.94	10.65




© 2001 Barton P. Miller

– 53 –

Security Issues

Limitations

- ❑ Can only ensure safety properties that can be expressed using tpestates + linear constraints
 - e.g., cannot handle nonlinear array subscripts
- ❑ Induction iteration method is incomplete
 - e.g., generalization capability is limited
- ❑ Limitations in handling of arrays
 - Lost precision
- ❑ Inherited limitations of static techniques
 - Must reject code that cannot be checked statically
 - Otherwise, there is the recovery problem


 © 2001 Barton P. Miller
 – 54 –
Security Issues

